DAM
Desarrollo de Aplicaciones Multiplataforma
2º Curso

# AD
# Acceso a Datos

UD 8
Programación de componentes
de acceso a datos
Parte 3b

IES BALMIS
Dpto Informática
Curso 2019-2020
Versión 1 (09/2019)

# 11. EJERCICIOS – API Rest sin BD

**UD8Ejer1101 – ApiRestMathAmpliado (2)**

Ampliar proyecto **ApiRestMathAmpliado** creado en los apuntes para que disponga de los métodos:

- **Restar**: también recibirá dos parámetros y devolverá el resuñtado de restar los dos parámetros recibidos. Ejemplo:
  - http://localhost:8081/math/apirest/aritmetica/restar/20/12
- **Multiplicar**: también recibirá dos parámetros y devolverá el resultado de restar los dos parámetros recibidos. Ejemplo:
  - http://localhost:8081/math/apirest/aritmetica/multiplicar/5/7

***Fuente**: https://www.programacion.com.py/web/java-web/web-service-rest-jersey-con-java-ee*

**UD8Ejer1102 – ApiRestFrutasKg (4)**

Partiendo del proyecto **ApiRestFrutas** crea el proyecto **ApiRestFrutasKg** que use la clase Frutas.

La url será:

http://localhost:8081/fruteria/recursos/frutas

Se desea añadir a la clase Frutas el campo **kg** de tipo int, con los kilos de stock que tenemos. Se puede observar que para que la respuesta por defecto de Java es en JSON, pero si se sea también en XML hay que realizar el mapeo de JAXB.

El ServiceRest ofrecerá los siguiente métodos produciendo y consumiendo tando JSON como XML:

- **GET => /frutas**
  - public List<Frutas> get()
- **GET => /frutas/{name}**
  - public Frutas get(@PathParam("name") String nombreFruta)
- **POST => /frutas**
  - public void post(Frutas fruit)
- **PUT => /frutas**
  - public void post(Frutas fruit)
- **DELETE => /frutas/{name}**
  - public void delete(@PathParam("name") String nombreFruta)

El servicio comprobará:

- en POST que el nombre de la fruta no existe ya en la lista antes de insertarlo.
- en PUT que el nombre de la fruta existe ya en la lista para actualizar los kg.
- en DELETE que el nombre de la fruta existe ya en la lista para aliminarlo.

## UD8Ejer1103 – ApiRestUsuarios (3)

Crea el proyecto **ApiRestUsuarios** que ofrecerá un servicio de API Rest en los formatos **JSON y XML** para acceder a la colección **usuarios** de la BD **apirest** de **MongoDB**.

La url será:

http://localhost:8081/apirest/recursos/usuarios

La base de datos se puede inicializar con los siguientes datos:

```
use apirest

db.usuarios.insert([
{
    id: "01",
    nombre: "Juan Soler",
    edad: 23,
    idioma: "Español"
},{
    id: "02",
    nombre: "Elena López",
    edad: 27,
    idioma: "Español"
},{
    id: "03",
    nombre: "George Smith ",
    edad: 19,
    idioma: "Inglés"
},{
    id: "04",
    nombre: "Harry Brown",
    edad: 25,
    idioma: "Inglés"
}])
```

Se deberá crear la clase Usuario para mapear el resultado a Java. Recuerda que tienes que añadir anotaciones JAXB para permitir usar XML.

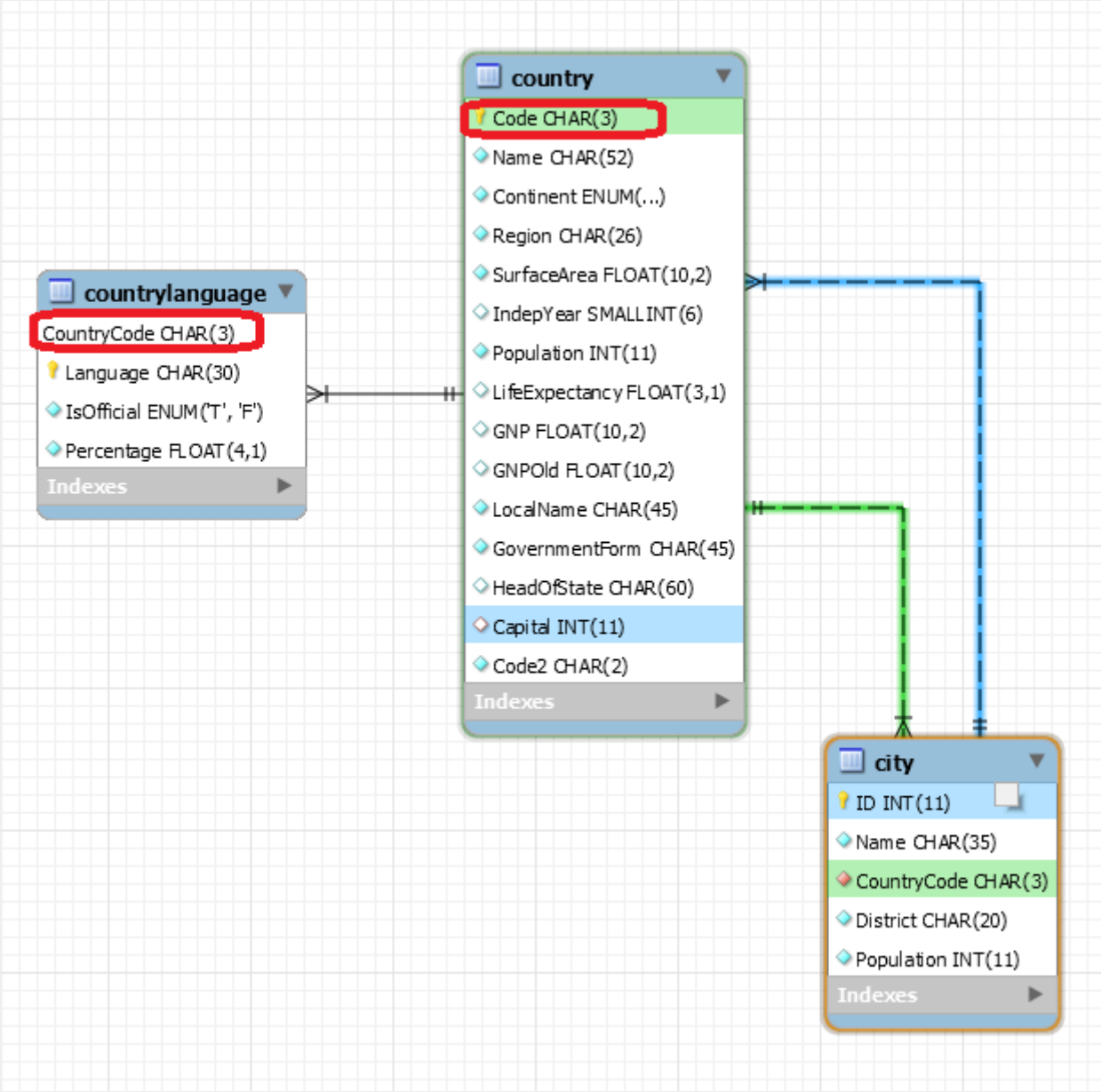
Los métodos a implementar son:

- **GET => /usuarios**
  - Devuelve los datos de todos los usuarios de la BD
- **GET => /usuarios/{id}**
  - Devuelve los datos del usuario cuyo id es el indicado
- **PUT => /usuarios**
  - Recibe en el body los datos de un usuario
  - Localiza y actualiza del usuario la edad y el idioma
  - El response que devuelve será
    - Si no lo encuentra
      - Status: NOT_FOUND
      - Mensaje: "Error: el usuario no existe"
    - Si lo actualiza
      - Status: NO_CONTENT
      - Body vacío
    - Si no lo actualiza
      - Status: BAD_REQUEST
      - Mensaje: "Error al actualizar usuario"

## UD8Ejer1104 – ApiRestWorld (2)

Tenemos una BD en MySQL denominada World. El profesor entregará el script para crearla.

El esquema relacional es:




Un país (country) tiene muchas ciudades (city), pero además tiene una capital (city).
En un país (country) se hablan muchas lenguas (contrylanguage).

Crea el proyecto **ApiRestWorld** que ofrecerá un servicio de API Rest en los formatos **JSON** para acceder a la BD **world** de **MySQL.**

La url será:

http://localhost:8081/world/XXXX

Para esto, el PATH de **ServiceRestWorld.java** será **@Path("")**

Se deberá crear la clase Usuario para mapear el resultado a Java. Recuerda que tienes que añadir anotaciones JAXB para permitir usar XML.

Los métodos a implementar son:

- **GET => /ciudad/{id}**
  - Devuelve los datos de la ciudad cuyo id es el indicado
  - Por elemplo:
    - http://localhost:8081/world/ciudad/3
- **GET => /pais/{code}**
  - Devuelve los datos del país cuyo code es el indicado
  - Por ejemplo:
    - http://localhost:8081/world/pais/FIN
  - Los campos a devolver en JSON son:
    - String **code**;
    - String **name**;
    - String **continent**;
    - Double **surfacearea**;
    - City **capital**;
    - ArrayList<City> **ciudades**;

# 12. EJERCICIOS – API Rest con JPA (Asistente)

**UD8Ejer1201 – ApiRestWorldDatabase (2)**

Crear el proyecto **ApiRestWorldDatabase** y genera un API Rest con el asistente "Restful Web Service from Database".

Para que funcione con UTF8 deberemos añadir en la conexión además de los parámetros conocidos, 3 más:

```
jdbc:mysql://localhost:3306/world?
zeroDateTimeBehavior=convertToNull&amp;
useSSL=false&amp;
autoReconnect=true&amp;
useUnicode=true&amp;
characterEncoding=UTF-8&amp;
characterSetResults=UTF-8
```

Para mostrar los datos de una ciudad o un pais tendremos:

```
/mysql/world/country/FIN
/mysql/world/city/4
```

Como countrylanguage tiene una clave compuesta, para indicar el id en la URL se indican los valores con sus nombres separados por ';':

```
/mysql/world/countrylanguage/id;countrycode=ABW;languagename=English
```

Añadiremos un nuevo método GET que mostrará todas las ciudades de un país con la URL:

```
/mysql/world/city/pais/{code}
```

**Para ello añadiremos:**

**en City.java la Query:**

```
@NamedQuery(name = "City.findAllCountry",
    query = "SELECT c FROM City c WHERE c.countrycode.code = :code")
```

**en AbstractFacade.java el método:**

```
public List<T> findAllCountry(String code) {
  javax.persistence.Query q =
     getEntityManager().createNamedQuery("City.findAllCountry");
  q.setParameter("code", code);
  return q.getResultList();
}
```

**en CityFacadeREST.java el método:**

```
@GET
@Path("/pais/{code}")
@Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
public List<City> findAllCountry(@PathParam("code") String code) {
  return super.findAllCountry(code);
}
```